ADV2-D60 **PATENT APPLICATION**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	Examiner: Mussa Shaarwat
NETWORKFAB CORPORATION	Group Art Unit: 2128
Serial No.: 09/611,793	
Filed: 7/7/2000	February 28, 2005
For: IMPROVED METHOD FOR COMMUNICATION WITH REAL-TIME REMOTE DEVICES OVER WIDE-AREA COMMUNICATIONS NETWORKS	San Diego, California 92108

DECLARATION OF Dmitri Soloviev under 37 CFR §1.131

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Dear Sir:

I, Dmitri Soloviev, declare that:

1. I am the inventor of the invention that is the subject of the instant application for patent;
2. I was asked by Karl M. Steins to prepare a letter regarding the circumstances and timing of the conception and reduction to practice of my invention; and
3. Attached hereto is a true and correct copy of the letter which I prepared as well as evidentiary support for my statement for the dates of conception and reduction to practice of the invention that is claimed in the subject Patent Application.

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements are made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 19 of the United States Code, and such willful

false statements may jeopardize the above-identified Application and any patent issuing therefrom.

Signed at Santa Clara, CA

on this day of February 28, 2005

By: _____

Print Name: Dmitri Soloviev.

February 28, 2005

To Whom It May Concern:

Subject: My Invention that is claimed by U.S. Patent Application No. 09/611,793

Claimed Invention

My invention entitled "Improved Method for Communication with Real-time Remote Devices over Wide-area Communications Networks" (hereinafter "subject invention") is disclosed and claimed by U.S. Patent Application serial number 09/611,793.

Pertinent Facts relating to Conception and Reduction to Practice

Attachment I is an invention disclosure prepared by myself and others soon after conception of the subject invention.

1. Conception date. I conceived of the subject invention on or before January 2, 2000; Attachment I reflects my notes made at the time to document the subject invention. On February 18, 2000, I participated in a telephone conference with Karl M. Steins (attorney of record for this application) during which we discussed the technical details of my invention.
2. Claimed invention. As first attested to by myself in the declaration filed with the patent application filed for the subject invention, the claims of application serial number 09/611,793 claim my invention.
3. Diligence and reduction to practice. From conception until before February 18, 2000, I continued to work on the subject invention until a working prototype was created and tested prior to February 18, 2000.
4. Constructive reduction to practice. Between February 18, 2000 and July 7, 2000, I worked

continuously with the attorney of record to complete and file Patent Application Serial Number 09/611,793.

Dmitri Soloviev

Attachment I: Invention Disclosure Document prepared for Karl M. Steins in January 2000

Inventors? Dima Soloviev
Nationality Canadian.
Date of Invention 11/2002

Improved method for communication with real-time remote devices over wide-area communications networks

[*copy the general discussion of remote devices here*]

1Definition of the problem

Following discussion is based on sample arrangement of devices shown on Fig.1:

Fig.1:

[COMPUTER] <————> network <——> [DEVICE]

Here *DEVICE* is a remotely operated device; it is connected to the *network* and is entirely controlled by *COMPUTER*.

The traditional approach [*insert patent ref here*] is to send device control commands or data in packets, one by one, over the network. These packets arrive to the device, get interpreted and executed in order they are received, and then results of execution are sent back to the controlling computer, again in natural order. For example:

COMPUTER ==> "Turn on the light" ==> DEVICE

COMPUTER <== "Light is now on" <== DEVICE

This method works well if functioning of the device does not depend on time of arrival of commands. Some devices do exhibit such real-time dependency. For example, let's consider a mechanical arm that applies a sticker to a box when the box is delivered on continuously moving conveyor belt:

COMPUTER ==> "Tell me when box comes" ==> DEVICE

...

COMPUTER <== "The box is in position" <== DEVICE

COMPUTER ==> "Apply sticker" ==> DEVICE

COMPUTER <== "Sticker applied" <== DEVICE

The device will not work if there is a significant delay between the moment when device notices the box in position and the moment when it receives a command from computer to apply a sticker. Such delay can be estimated as

(1.1) $delay = 2 * network\_delay$

where *network_delay* is the time required to pass the message from device to computer or back. If the network has non-deterministic delays then this formula will be based on statistics and we have to discuss probability of some specific delay.

Most modern networks have non-deterministic nature [*re: Ethernet, TCP/IP*] and therefore do not allow to predict exact delay. Random variances of time passed between the moment the device detects the box in position and the moment it receives a command to apply a sticker will result in sticker applied to random spots on the box. These spots will be centered around certain point on the box; distance between this

statistically calculated point and the originally intended point will be proportional to average network delay and the conveyor belt speed.

2Compensating for average network delay

Assuming that these two elements do not significantly change, the device could compensate for this relatively stable error by signaling the presence of the box before it actually reaches the position. During *delay* time (see 1.1 above) the box continues moving, and when the command from computer finally arrives the box is in correct position and sticker can be now applied properly.

This solution does not always work perfectly because average network delay can change over larger periods of time. For example, the network may be more loaded at day time than at night. This will require some adjustments, as well as method of measuring the *delay* value at run time.

The described compensation mechanism relies upon prediction of future event (such as "box in position"). In many cases such prediction is not possible. For example, photo camera set up to take pictures of lightning storm can not predict the lightning, so delay between detection of the lightning and taking a picture may cause the picture being taken after the energy discharge ends, making the result useless.

3Compensating for random network delay

Even if the device is capable to compensate for average, relatively constant *delay* value, it can not predict additional random delays which occur on packet-to-packet basis. The corrected version of formula (1.1) will look like this:

(1.2) $\quad delay = 2 * average\_delay + random\_delay() + random\_delay()$

where *average_delay* is the average delay of one-way packet, and *random_delay()* is a function that returns a random delay of a packet on its way to computer or back to the device. Each value returned by *random_delay()* has a known chance (probability) to fall within certain range. Exact distribution of chances depends on characteristics of the network.

The device can not predict the value that *random_delay()* will return, and therefore it will be erroneously applying stickers around the correct spot (if compensation for the average network delay is done correctly).

4More examples of real-time devices

The scenario discussed above applies not only to labeling equipment but generally to any device that provides or requires feedback from controlling computer. Even a telephone call via geostationary or high-orbit satellite introduces delay as large as one or more seconds; it makes it difficult sometimes to speakers to "synchronize". Fast devices are more vulnerable to unexpected delays. Video cameras, for example, capture frames at very precise moments, and computer must read the data within very short period of time. Any delays will invalidate the capture process and video stream becomes defective - either some frames are lost or entire process goes out of synchronization, causing severe malfunction.

5Distributed decision-making process

We propose to move some functionality originally provided by COMPUTER to the DEVICE itself. This functionality will ensure that real-time decisions will be carried out locally, within the DEVICE itself, without aid of the COMPUTER. Only the absolutely required portion of functionality should be provided by DEVICE; all other functionality remains implemented by COMPUTER. The DEVICE still maintains two-way connection to the COMPUTER and informs COMPUTER about decisions made locally, so that COMPUTER has at all times an accurate knowledge of device state. The DEVICE does not need to depend on exact time when the COMPUTER receives the information from the DEVICE, and therefore can work faster and more reliably.

The following scenario involves the box-labeling device (see examples above) that has internally implemented function: "apply sticker when box is in position":

COMPUTER ==> "Tell me when box comes, apply sticker when ready" ==> DEVICE

...

COMPUTER <== "The box is in position, sticker applied" <== DEVICE

COMPUTER ==> "Tell me when another box comes, apply sticker when ready" ==> DEVICE

This scenario illustrates that device makes its own decision when to apply the sticker. However high-level decisions, such as what sticker to apply, what sort of boxes to expect, when to start or stop working, etc. are made by controlling computer. This allows centralized control and also reduces complexity of the device.

6Use of devices that were designed to be connected locally as remote devices

Most of existing and newly developed devices are intended to be operated (controlled) by local computer. The way these devices are manufactured and typically connected to the controlling computer does not allow remote connection, via a network. These devices expect to receive all commands from computer virtually instantly and they expect the computer to receive everything that device sends also virtually instantly. As shown above, such devices may malfunction if they are used as remote devices as described below.

7Known methods of connecting a device that was designed to be connected locally to function as a remote device

There is specialized equipment that presents to the DEVICE the interface that looks electrically and mechanically exactly like COMPUTER; similarly, such equipment presents the COMPUTER with an interface that looks exactly like DEVICE.

[*insert picture of generalized Host-OurBox-Network-OurBox-Device here*]

In some cases it is possible to implement COMPUTER side of the equipment in COMPUTER itself, if COMPUTER already has suitable interface to the NETWORK:

[*insert picture of simplified Host/Driver-Network-OurBox-Device here*]

In both cases the intermediate equipment transparently passes commands and data

to and from DEVICE. This makes such equipment "invisible" to the COMPUTER and therefore COMPUTER can still run existing software unchanged, even if DEVICE is now remotely connected. This is an important advantage of such "transparent" solution *[insert reference to patent ## here]*

However the method described above is vulnerable to network delays. As shown above, such delays can severely disrupt the functioning of the DEVICE even if all commands are delivered correctly but at wrong time. This method does not provide any means to compensate for random network delays and therefore may occasionally fail with certain devices, on certain networks, under certain circumstances.

This method also does not provide means for adjusting the volume of data traffic generated by COMPUTER or DEVICE to fit the available bandwidth of the network. Typically local hardware connection (cable) allows much faster data rate than a commonly accessible network. This method does not allow to degrade the performance of the device gradually; instead, the complete malfunction will result if some of the data can not be transferred to or from the DEVICE.

8How existing remoting method works

The equipment [*ChuBox*] used to connect devices over the network usually consists of a hardware interface that is compatible with DEVICE, and a specialized computer that receives all data that device sends, wraps them in transport packets and sends over the network. When response packets arrive, this specialized computer extracts data and sends it to the device via the hardware interface.

Similar equipment has to be employed on other end of the connection. The COMPUTER connects to it using the hardware interface. A simplified version of the equipment may utilize COMPUTER itself; in such case COMPUTER connects to the device using simulated hardware interface, so that all existing software that accesses the DEVICE does not need to be altered.

The described equipment is device-independent, as long as the hardware interface remains suitable for connection of the device. The data that gets transferred over the network is only encapsulated (to provide security and reliability) but not altered in any way.

9How to improve existing method to support remote real-time devices

In order to support real-time devices we propose to enhance functionality of the DEVICE-connected part of remoting equipment (DCPRE) so that it becomes device-aware and can perform certain (time-sensitive) procedures on connected DEVICE on its own, without specific instructions coming from controlling COMPUTER. Similarly, the COMPUTER-connected part of remoting equipment (CCPRE) supports device-aware DCPRE by sending only asynchronous, non-time-sensitive commands to the peer, along with instructions how to act if a time-sensitive event occurs.

Such pair of connected CCPRE and DCPRE will use data protocol somewhat different from the "raw" data protocol utilized by traditional remoting equipment. This "smart" protocol will be constructed to eliminate time-sensitive instructions, but instead the DCPRE will be performing such time-sensitive procedures on its own, manipulating the DEVICE in best possible way without assistance from the

COMPUTER or CCPRE.

In order to remain "transparent" to COMPUTER-based device drivers and software, the CCPRE will translate the device state changes that DCPRE sends over the network and will apply those changes to a model of the DEVICE simulated within the CCPRE. Then all data sent to the remoted DEVICE will be executed locally, within the CCPRE, using the modeled DEVICE; at the same time if an update needs to be sent to the actual DEVICE the CCPRE will generate appropriate request in "raw" or "smart" format and will send it to DCPRE to be carried out. Alternatively, the CCPRE may choose a traditional method for certain commands; then the data is sent directly to DCPRE (and DEVICE) and CCPRE waits until the DEVICE and DCPRE respond.

Therefore, both DEVICE and COMPUTER become connected to simulated model of the expected peer (COMPUTER and DEVICE accordingly). The DEVICE gets its instructions from DCPRE as if the COMPUTER generated them, though the actual COMPUTER does not know about the time-sensitive event yet. Similarly, the COMPUTER does not access DEVICE directly, but instead interacts with modeled device inside of CCPRE which responds appropriately to COMPUTER's commands.

Two modeled objects - a DEVICE simulated within CCPRE and COMPUTER simulated within DCPRE - are synchronized over the network using protocols most suitable for the purpose. These protocols may implement data compression, enhancement, encryption and other processing that improves the quality of remoting. For example, remoted video camera generates large amount of "raw" data. DCPRE and CCPRE may use video compression protocols between them, so that necessary network bandwidth remains within acceptable limits. The video camera still generates "raw" data, and COMPUTER still receives "raw" data decompressed and placed into the model of the camera by DCPRE. Delays in synchronization between models (caused by network latency) do not directly affect COMPUTER or DEVICE because models are programmed to behave exactly as "real" COMPUTER or DEVICE would behave.

The described approach provides means for "transparent" remoting of real-time devices. It requires additional software in DCPRE and CCPRE. This software is highly device-dependent because it has to maintain a model of the DEVICE in CCPRE; the DCPRE also has to have device-dependent software which understands model synchronization protocol (so that it can update the model state in CCPRE); additionally, this DCPRE-based software has to be able to perform certain functions that can not (or should not) be performed directly by remote peer (such as real-time procedures).

This newly proposed method allows to connect DEVICE via networks with varying bandwidth. The network performance only affects the speed of synchronization between model of DEVICE and the actual DEVICE. If the network is slow then the model of DEVICE (maintained in CCPRE) will be updated less often; similarly, the simplified model of COMPUTER (maintained in DCPRE) will be filled with new data less often; the DCPRE-based device-specific model can fill the gaps with locally generated data.

This method can also allow sharing of a DEVICE between several COMPUTERs. Such sharing becomes possible if the DEVICE itself by its nature and design can be accessed by several COMPUTERs simultaneously, and if application and driver

software in each COMPUTER is sufficiently well designed to accommodate such sharing. Each COMPUTER connects to its own CCPRE, and each CCPRE connects to one (shared) DCPRE that is connected to one (shared) DEVICE. The DCPRE can be programmed to update device models in all connected CCPRE. All COMPUTERs will be interacting with their own models of the device (located in connected CCPRE). Only one (controlling) CCPRE will be allowed to alter settings of the device; alternatively, any CCPRE can alter settings of DEVICE and these changes will then propagate to all connected CCPRE. In former case non-controlling CCPRE may be able to control certain settings of the modeled device, as long as this CCPRE can convert the data (that comes in form of state updates from shared DCPRE) into the format appropriate for the requested state. As an option, all CCPRE except the controlling CCPRE may lock settings of their models to match settings used by controlling CCPRE. Then all COMPUTERs will be working with same models. If any CCPRE is allowed to change DEVICE settings then all CCPRE must be capable to propagate new settings back to drivers and application software running on COMPUTER; this is often not supported by existing applications.

10Example 1: Remote video camera

The video camera receives a request "Start". Within short time several packets of video data are produced. Computer must immediately issue same number of requests "Read" to read the data. If computer is too late with its "Read" requests, first *N* packets of data from the camera will be lost, and last *N* requests from the computer will not be satisfied (causing garbled image or computer lock-up, depending on driver or application software running in the computer).

The CCPRE-based model maintains a copy of video screen. When computer issues request "Start" this request is passed to DCPRE, along with instructions to read certain number of packets and send them back to DCPRE when convenient. Upon arrival from DCPRE, these packets will be applied to CCPRE-based copy of video screen. When computer issues "Read" requests the CCPRE-based model returns the data that is currently in the copy of video screen.

11Example 2: Remote digital audio speakers

Digital audio speakers receive continuous and consistent stream of data packets from the computer. Every packet represents a short fragment of audio signal. When packets arrive as required, one by one, at precise time, the audio fragments join seamlessly and speakers produce continuous audio signal of good quality. If packets do not arrive with constant rate but instead some come later and some come earlier, all missing packets will be heard as "clicks" or noise, and if several packets arrive together most of them will be discarded by speakers (because some of them are too late, and other are too early and speakers don't have enough internal memory to store them for later).

The DCPRE-based model of computer maintains a large buffer for digitized audio packets. CCPRE does not need to have a model of speakers because speakers do not produce time-sensitive events; instead, CCPRE will send everything it receives from actual COMPUTER to the DCPRE. DCPRE will queue arriving audio packets into the buffer, but instructs speakers to start playing sound only when buffer is optimally full. From CCPRE (and COMPUTER) point of view, speakers started playing sound

immediately. However the audio data was only queued in DCPRE. When speakers are told to start playing, DCPRE will be sending audio packets from the queue, at precise time they are expected by speakers. If CCPRE (and COMPUTER) fail to supply enough audio data via the network (due to network delays) then DCPRE will be still sending to speakers previously queued packets. The length of queue will be chosen to cover most of random network delays. If the network delay is larger then DCPRE runs out of data. In such case DCPRE may start sending its own packets made to mask the gap in audio; alternatively (as an example), DCPRE can instruct speakers to lower the volume when queue is about to become empty, so when all data from queue is sent the speakers are already muted and no "click" is heard. There could be many other methods to mask the loss of audio data, depending on requirements and features of speakers.

ADV2-D60 **PATENT APPLICATION**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	Examiner: Mussa Shaawat
NETWORKFAB CORPORATION	Group Art Unit: 2128
Serial No.: 09/611,793	
Filed: 7/7/2000	February 28, 2005
For: IMPROVED METHOD FOR COMMUNICATION WITH REAL-TIME REMOTE DEVICES OVER WIDE-AREA COMMUNICATIONS NETWORKS	San Diego, California 92108

DECLARATION OF Rick Lu under 37 CFR §1.131

Honorable Commissioner of Patents and Trademarks
Washington, D.C. 20231

Dear Sir:

I, Rick Lu, declare that:

1. I am the president of Networkfab Corporation, Assignee of the subject patent application;
2. I have reviewed the "Declaration of Dmitri Soloviev under 37 CFR §1.131", and declare that I am a witness to each and every fact attested to by Dmitri Soloviev in his declaration; and
3. I declare that the facts attested to by Dmitri Soloviev in his Declaration, to the best of my recollection and notes, accurately reflect the dates and circumstances related to the conception and reduction to practice of the invention that is the subject of patent application serial number 09/611,793.

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements are made with the knowledge that willful false statements and the like are punishable by fine or

imprisonment, or both, under Section 1001 of Title 19 of the United States Code, and such willful false statements may jeopardize the above-identified Application and any patent issuing therefrom.

Signed at Santa Clara, CA

By: Rick Lu

on this day of February 28, 2005

Print Name: Rick Lu